A Stab at A2 Criteria

[redacted]
and
GEORGE F. JELEN

Since 1983, when the DOD Computer Security Center published what has come to be known as the Orange Book, the computer security world has recognized only seven classes of security protection. This paper represents the authors' proposal for an eighth – Class A2. The paper presents the changes being proposed and the rationale behind them, and the appendix contains the suggested A2 criteria. The major intent of the proposed additions to the present A1 criteria is to attempt to deal more directly with the issues of data integrity and the threat of subversion.

INTRODUCTION

In August 1983, what was then called the Department of Defense Computer Security Center published a document entitled *Department of Defense Trusted Computer System Evaluation Criteria.* In December 1985 the *Criteria* was republished as a DOD standard. The covers of both editions were orange, and the document quickly came to be known within computer security circles as "the Orange Book."

The evaluation criteria spelled out in these documents classify computer products into seven hierarchical classes of security protection. The seven classes, in order of increasing protection, are D, C1, C2, B1, B2, B3, and A1. The *Criteria* provides for classes beyond A1 but lists requirements for such classes only in the sketchiest of terms. The appendix to this paper represents our stab at the next class, A2, and the paper itself explains the changes we are proposing and why.

Our purpose in offering this paper is to stimulate and to focus discussion within the computer security community on the logical next steps beyond A1 and to direct research and development along specific lines that we believe are essential. We believe that the time is ripe to begin a serious dialogue regarding an A2 product, but serious work cannot begin unless there is some indication of the directions that A2 developments should take.

In any discussion of additions or modifications to the Orange Book, two approaches could be taken. The first is to follow and preserve the current scheme. Under this scheme, the current requirements of all the lower criteria classes would be left unchanged, and any extensions of the current requirements would not appear until the A2 level. The second alternative is to backfill any new requirements (e.g., integrity, denial of service) into the current structure. This would entail adding requirements to already existing classes through A1 – in other words, a major rewrite. Although not necessarily convinced that a major rewrite might not have been better, we have avoided it by adopting the first

approach. We have preserved the principle of the current criteria that every succeeding criteria class and division include the requirements of lower classes and divisions, and we have tried to make the jump to A2 reasonably consistent with the size of previous steps between adjacent classes. We have therefore intentionally not included at the A2 level every improvement we could think of. Our intention is to initiate discussion of what is the reasonable next step for the criteria and have left plenty of room for additional classes beyond A2.

Our proposed A2 criteria differ from the present A1 criteria in at least two significant ways. First, they begin to deal with data integrity protection by introducing trust labels and by requiring tight binding of an object to its label; and second, they try to address more forthrightly the subversion threat by requiring development in a secure environment and by extending verification to the source code or higher order language (HOL) level.

The present Orange Book is based upon the precept that security, like any quality, must be designed in; it cannot be "bolted on." Developing a computer system to B2 or higher criteria requires careful attention to the base hardware and software architecture. The architectural approach required by B2 and the above systems cannot be achieved by "band-aiding" some set of additional features. It can be achieved only through a process that begins with early design and actively involves those participating in design and implementation of the lowest level aspects of the system. One implication of this observation is that products will come from the vendors – from industry. We have remained faithful to such reasoning. It is our view that A2 products will also have to come from industry, although probably with some cost sharing on the part of the government and perhaps with some marketing restrictions. We expect that products beyond A1 may incorporate some classified techniques or may be subject to export control restrictions. We have not required that any technology necessary to satisfy A2 must be available without restriction, as was generally assumed in lower criteria classes. Clearly, this will have an effect on cost; those who need the protections offered by A2 and above may have to be willing to pay a premium for the capabilities provided.

Requirements derive from policy. Each requirement in the present Orange Book is an instantiation of one of the "control objectives"[1] which, in turn, are derived from statements of national policy. Among the primary reasons for not including in lower criteria classes requirements that dealt directly with issues of integrity and denial-of-service is that there was, at the time the *Criteria* was issued, no national security policy dealing with these issues.

The *Criteria* requirements reflect achievable technology. At the time of the writing of the Orange Book, the "sanity check" that was applied to each requirement to be included was that there be at least three worked examples of the technique or mechanism in

1. See U.S. Department of Defense, *Department of Defense Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, December 1985, pp. 57–63.

question. The primary audience for the Orange Book was the vendors, and it was seen as unfair to ask a vendor to undertake a research activity in the pursuit of a commercial product.

We believe that both of these principles are reasonable and should continue to apply, at least at the A2 level. In fact, the second of these is seen to be a prime motivator for this paper. If the features and assurances described herein are deemed desirable, then there must be R&D activity, where necessary, directed toward providing the requisite theoretical solutions and worked examples. In the case of policy, we have stretched the rule somewhat and included requirements for which, while policy may not exist, a broad-based need has been articulated. Specifically, the authors feel that there has been a clear call for mechanisms that deal with both integrity and denial-of-service. Our current statement of the A2 criteria has a "place holder" for integrity mechanisms (the A2 proposal refers to "trust level") but carefully avoids mandating a specific definition or policy for integrity. Clearly, it will be necessary to narrow down the definitions, probably based upon existing laws and regulations, to a reasonable and widely applicable few. However, we envision that any integrity mechanism acceptable for trusted products will be rule-based, i.e., given some initial trust level and a series of state changes, the final trust level is always unambiguously determinable.

ASSURANCE

The direction of the *Criteria* is toward greater levels of assurance – the confidence that the intended security mechanisms are truly in place, remain in existence, work correctly, and are uncircumventable. Assurance is primarily derived from attention to architecture, design, design analysis, internal structure, and testing. Our proposed A2 criteria continue this thrust with requirements in several of these areas, borrowing heavily from techniques that have been standard fare in cryptography for years. Specifically, we extend formal techniques another level of abstraction from that currently required at A1. Where the A1 criteria ask that formal methods be applied to the formal top-level specifications (FTLS), we extend those techniques to the source code level, requiring that "a combination of formal and informal techniques . . . be used to show that the source code is consistent with the FTLS or with the model." Informal mapping of source code to object code is also required. Assurance techniques are even extended to tools for developing the task control block (TCB) (e.g., compilers, assemblers), which must be shown to generate correct code.

Assurance techniques are also specifically applied to hardware. While the current A1 criteria are not entirely silent on the subject of the hardware, we include at A2 very strong, albeit informal, analysis of the potential failure modes of the hardware. In addition, we ask for mechanisms to detect or preclude failure of critical TCB hardware elements. Finally, we propose that hardware reliability be addressed through design, analysis, and testing.

INTEGRITY PROTECTION

From the very beginning there has been little attention given within the computer security community to the data integrity problem per se.[2] There are probably several reasons for this. One of the factors that allowed progress within the DOD on computer security is that there is a consistent, explicit, and well-understood policy for protection against compromise. Everyone within the national security community was comfortable with the idea of classifications and clearances. The Bell and LaPadula formal security model was but a formal expression of these well-understood and accepted concepts. There is no analogous policy for integrity protection. Second, perhaps partly because there is no explicit policy, there seems to exist the belief that the integrity problem is somehow of lesser importance than the security problem. And third, since no one yet claims to have solved the security problem, and the integrity problem is even harder, few have appeared ready to tackle the more difficult problem. One of the results of all this is that there have been so few papers written on the subject that there has not yet evolved even a standard terminology with which to discuss the problem.

Through *Criteria* Class A1, integrity protection is required only to the extent that it is necessary for protection against disclosure. We believe that a system at the A2 level should offer integrity protection for its own sake. At the same time, we are aware that there are many different notions or models of integrity protection.[3] Willing neither to settle upon and mandate a particular notion or model nor to ignore integrity protection completely, we have opted for mandating the "hooks" that would permit integrity protection but have allowed the vendor to establish and enforce his own policy and model.

In our proposed criteria, data integrity is addressed by requiring the vendor to define and properly support an integrity policy of his choosing. We specifically state that the policy need not be a dual of the Mandatory Access Control policy, i.e., implement a partially ordered lattice. However, we require that any access control rules be clearly specified and the implications (e.g., labeling of subjects and objects) of the policy be identified. It must also be demonstrated that the implementation enforces the rules.

The requirement for mechanisms that enforce some integrity policy will impact those criteria dealing with labels, their protection, exportation, auditing, and perhaps the notion

2. Because no real disclosure protection is possible without it, the current *Criteria* pays some attention to integrity concerns. For instance, the DAC requirements throughout the *Criteria* enforce discretionary access control for modification as well as for disclosure. Additionally, the "system integrity" requirements, in conjunction with the assurance and testing requirements, support what could be referred to as "process integrity," through which trust in the correct and efficacious operation of the system is engendered.

3. Among the more interesting of the recent investigations into this area are the papers dealing with what has come to be known as the "Clark-Wilson Model." The defining papers are David D. Clark and David R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," *Proceedings, 1987 IEEE Symposium on Security and Privacy*, Washington, D.C.: Computer Society Press of the IEEE, 1987, pp. 184–94 and David D. Clark and David R.Wilson, "Evolution of A Model for Computer Integrity," *Proceedings of the 11th National Computer Security Conference: a Postscript*, Fort George G. Meade, Maryland: National Computer Security Center, 1988.

of single level and multilevel devices. However, it is not clear that all reasonable integrity policies will have the same implications in all these areas. Thus, while we have added requirements where the implications of trust labels were clear, we have remained silent in those paragraphs in which the implication was less clear. For example, one might envision an integrity policy in which the trust label represented the trustworthiness of the source of the data. The policy rules would preserve the label if and only if no modifications or additions were made to the object. The policy need not prohibit modifications to the original object but would automatically "downgrade" the label if the object were changed in any way since the original trustworthiness could no longer be guaranteed. In such a policy, a multilevel (in an integrity sense) device may have no meaning comparable to the present one, since the policy is not an access control policy. It may be reasonable to allow any and all devices to handle such data, since the label in question reflects only the believability or accuracy of the contents. That is, while there will be rules enforced for the handling of such labels, there need not be any notion of any hierarchy of labels, and there need not be any particular relation defined among labels.

In summary, we would require that any integrity policy be rule-based – that the rules be clearly defined and shown to be enforced. Additionally, we believe that it must always be possible to reason about the state of the system and the nature of the object to which the trust label is appended.

Trust Labels

As part of our increased focus on integrity protection, our proposed criteria require trust labels. We considered calling them "integrity labels," but we realized that it makes little sense to expend much effort to protect data from modification if the data we are protecting was invalid or untrustworthy in the first place.

Much of what is currently written about the integrity problem deals only with the sanctity of an object after origination and ignores the problem of the reliability or validity of the object when originated. If we are going to force ourselves to worry about the resistance to change (i.e., integrity) of some object, we ought also to make an equal effort to ensure that the object we are so protecting is initially valid and correct. In order for data to be worthy of trust or belief, they must have come from a reliable source and they must be preserved from inadvertent or deliberate modification.

Any comprehensive policy for integrity protection, therefore, cannot afford to ignore the problem of origination. How much care we take to protect a given datum element from modification ought to depend not only on how damaging it could be if it were modified, but also on how sure we are that it is correct or valid to begin with. The concept of trustworthiness, which we introduce, therefore embodies both the notion of validity at origination as well as protection from modification.

Labels, under our proposed criteria, are used to represent the relative trustworthiness of the subjects and objects with which they are associated. When data are exported, external labels, which correspond to the internal labels, must accompany the data.

The idea that some data are more believable than some others is certainly not counterintuitive. Certainly, too, there are ways of providing more or less protection from modification. The number of bits in a redundancy check scheme is an example of measurable integrity protection.

As with security or sensitivity labels, these trust labels must be associated with all subjects and storage objects in the system. Two methods for achieving integrity protection are through cryptographic sealing and through redundancy error detection and correction. Although these two methods can be employed in isolation, they are most effective when used together. Nevertheless, our proposed criteria are permissive in this regard and require only that "mechanisms be provided . . . that provide tight binding of the label and the storage object to which the label applies."

Exportation To Devices

Our proposed criteria require that when objects are exported to an external I/O device, the trust labels associated with that object shall also be exported and shall reside on the same physical medium as the exported information and shall be in the same form – either machine readable or human readable. With respect to trust labels, it makes no difference whether the device is single level or multilevel. This is a difference between sensitivity labels and trust labels. Even for single-level devices, when the TCB exports or imports an object over a communication channel, the protocol used on that channel is required to provide for the unambiguous pairing between the sensitivity and trust labels and the associated information that is sent or received. By contrast, single-level I/O devices and single-level communication channels are not required to maintain the sensitivity labels of the information they process.

System Integrity

With respect to system integrity, our proposed criteria require that hardware or software features be provided that can be used periodically to validate the correct operation of the on-site hardware and firmware elements of the TCB. Additionally, some set of such features must be available to operate in background mode, such that the TCB is continually being checked while on line. It must be possible for the system administrator to disable these background check functions dynamically, but such an action will be audited.

Subversion

There are three large classes of threat to the security of a computer system. In order of severity they are inadvertent mishap, deliberate penetration, and subversion. The

ascending criteria classes of the present Orange Book can be viewed as a way of successively dealing with these different classes of threat. *Criteria* classes C1, C2, and B1 offer increasing protection against inadvertent mishap but offer little protection against penetration. Classes B2, B3, and A1 offer increasing protection against penetration but, with the exception of the trusted distribution requirement introduced at the A1 level, offer very little protection against subversion. We believe that at the A2 level some protection against subversion must be offered, and this conviction has been a major consideration guiding our formulation.

Computer subversion is the name generally applied to deliberate, malicious modification of executable code or hardware within a computer system. The subverter changes the system so that it is no longer the system intended. The subversion can be accomplished at any time during the system's life from the earliest stages of design to the last day of its use. A component or system could be subverted either to permit later penetration or to serve as an act of sabotage – to nullify or to degrade the system's capability. Forms of subversion include the trap door, the Trojan Horse, and computer viruses. A trap door is a software device activated by some preset sequence of characters that is input to the computer, usually for the purpose of circumventing the system's security controls. A Trojan Horse is an artifice, usually a program, that has both an overt and a covert function. The overt function, often called the "lure," typically serves some often used and highly useful purpose. The covert function, which executes concurrently with the overt function, is malicious and performs the subverting act. Computer viruses are programs that are able to attach themselves to other programs and cause these newly "infected" programs to become viruses as well. Computer viruses can be used to propagate and implant trap doors or Trojan Horses.[4]

Subversion is attractive because it is virtually undetectable, usually permanent and not that hard to do. The subverted component could be part of the original hardware, part of a modification or update, or the result of replacement during maintenance. Thus, it can be done at any time during a system's life from the earliest stages of design to the last day of use. The opportunities are endless, and the subverter need succeed only once. Once subverted, the component remains compromised forever. For these same reasons, protecting against subversion is quite difficult. Yet to ignore it, we believe, would be sheer folly.

Several mechanisms in the complete A2 criteria as we propose them (i.e., code verification, configuration management, controlled environment, and trusted distribution) offer some protection against subversion. Code verification is helpful because it raises the level of assurance that the code does exactly what was intended and nothing

4. For a thorough introduction to the subject of subversion, see Philip A. Myers, "Subversion: The Neglected Aspect of Computer Security," Masters Thesis, Naval Postgraduate School, Monterey, California, June 1980. For a discussion of computer viruses in particular, see Fred Cohen, "Computer Viruses: Theory and Experiments," 7th DOD/NBS Computer Security Conference, Fort George G. Meade, Maryland: DOD Computer Security Center, 1984.

else. Configuration management places controls on changes, a common object of subversion efforts. A controlled development environment reduces the risk of subversion by requiring that everyone who participates in the development of any hardware or software for the system be vetted or investigated in some way. The DOD clearance process would be one way, but there could be others.[5] Finally, a trusted distribution system, also required for *Criteria* Classes B3 and A1, helps to protect the computer from subversion during transport.

Code Verification

For certification at the A1 level, the *Criteria* requires a formal model, a descriptive top-level specification (DTLS), and FTLS. The A1 criteria permit a combination of formal and informal techniques to show that the FTLS is consistent with the model. They also require a mapping of the FTLS to the source code of the TCB.

At the A2 level we would disallow informal techniques in the FTLS-to-model mapping and require that only formal techniques be employed. And whereas at the A1 level only informal techniques were required to show that the TCB implementation (i.e., in hardware, firmware, and software) is consistent with the FTLS, at the A2 level we are requiring a combination of formal and informal methods. We further propose extending the mapping process to the object code level by requiring a source-to-object code mapping to provide evidence of correct implementation. And we require that any tools used in the TCB development (e.g., compilers, assemblers, loaders) be shown to generate correct code.

Configuration Management

At Class B2 and above the Orange Book requires that a configuration management system be in place to control changes to specifications, documentation, and the code itself. At the A1 level the configuration management system is extended to all security-relevant hardware, firmware, and software as well as to all formalisms. Our proposed A2 criteria require that the configuration management system also be applied to all tools (e.g., compilers, assemblers, loaders) used for generating TCB code. The configuration management system is intended to assure a consistent mapping among all documentation and code associated with the current version of the TCB.

Trusted Design Environment

The *Criteria* is seen as largely addressing threats posed by system users – those with at least some legitimate access to the system and its resources. Configuration management,

5. For a more thorough discussion of the efficacy of a DOD clearance as a way of mitigating the threat of subversion, see George F. Jelen, "Information Security: An Elusive Goal," *Program on Information Resources Policy*, Harvard University, Cambridge, Massachusetts, Publication P-85-8, June 1985, pp. III-46–49.

in addition to its obvious value in controlling changes, also supplies techniques that begin to address the issue of hostile developers and substitution attacks. At A1, the trusted distribution requirement clearly recognizes such a threat by demanding some form of vendor-to-recipient authentication. Our proposed A2 criteria, besides imposing more extensive configuration control, also extend this thrust by requiring a trusted development environment. Specifically, our criteria require the TCB to be developed in a trusted facility with only trusted (i.e., cleared) personnel.

Other Changes

In addition to the above changes to support data integrity protection and help deal with subversion, we are proposing other changes to upgrade the overall quality of A2 products. They include strengthening of the requirements for Object Reuse, Identification and Authentication (I&A), Trusted Path, Trusted Facility Management, and Security Testing.

Object Reuse

Our proposed A2 criteria tighten up the requirements attendant to the reuse of storage objects. The requirement of lower classes, imposed at the C2 level and unaltered through Class A1, is only that previous authorizations to information within a given storage object must be revoked before that object is again used for storage. Since the data stored in the storage object are still resident, the present criteria leave open the possibility for scavenging through the storage object until the object is reallocated. Under the change that we are proposing for the A2 criteria, objects that are released must be cleared immediately upon deallocation.[6]

Identification and Authentication

Our proposed A2 criteria strengthen the I&A requirement by requiring the capability to accept and compare at least two independent personal identifiers, such as a password and a biometric. Additionally, trusted networking is specifically supported through the requirement that I&A information be capable of being forwarded to a foreign host.

Trusted Path

Another change that we are suggesting deals with the notion of trusted path. In our proposed criteria we tighten the trusted path specification to help cope with the intelligent terminal problem. The proposed criteria require that the connection between the user and

6. This requirement clearly has potential performance impact. While we recognize that such a requirement might be eased for practical considerations, we have chosen to take the more conservative security approach for this proposal.

the TCB be established and verified regardless of intervening hardware or software. Although they are not mandated, we expect that encryption techniques will be needed to satisfy this requirement.

Trusted Facility Management

Yet another change that we are suggesting would toughen the Trusted Facility Management criteria by requiring them to support the option of "two-man control" or split knowledge. The present criteria for Classes B3 and higher require separate operator and administrator functions and attempt to constrain administrator functions to a minimum number of auditable actions. We believe that, for systems requiring protection at the A2 level, it should be possible to further constrain the system such that any or all of the administrator and operator functions require the coordination of two persons.

Security Testing

Our proposed A2 criteria strengthen the security testing requirement in two ways. Present A1 criteria require that design documentation, source code, and object code be subjected to thorough analysis and testing by a team that understands the specific implementation of the TCB. In our proposed A2 criteria, we add hardware to the list of things that require testing. We also impose a new requirement for failure mode analysis – an analysis of potential hardware or software failures to determine if any of them could result in undetected security policy violations. In addition, a proposed addition to the system integrity criterion requires that sufficient hardware or software mechanisms be provided to ensure that a failure of the TCB hardware cannot go undetected without the simultaneous failure of at least two independent elements.

SUMMARY

As was pointed out in the introduction, our proposed A2 criteria differ from the present A1 criteria in at least two significant ways. First, they begin to deal with data integrity protection by introducing trust labels and by tight binding of an object to its label; and second, they try to address more forthrightly the subversion threat by requiring development in a secure environment and by extending verification to the source code or higher order language (HOL) level.

We realize that the policy that guided the writing of the Orange Book was to include nothing that had not already been successfully built. We believe that this is a reasonable approach for these criteria as well. Thus, before these criteria are promulgated officially, research needs to be directed toward solving the challenges they offer and demonstrating worked examples. Therefore, we do not believe that it is too early to begin considering what features and assurances ought to comprise an A2 system.

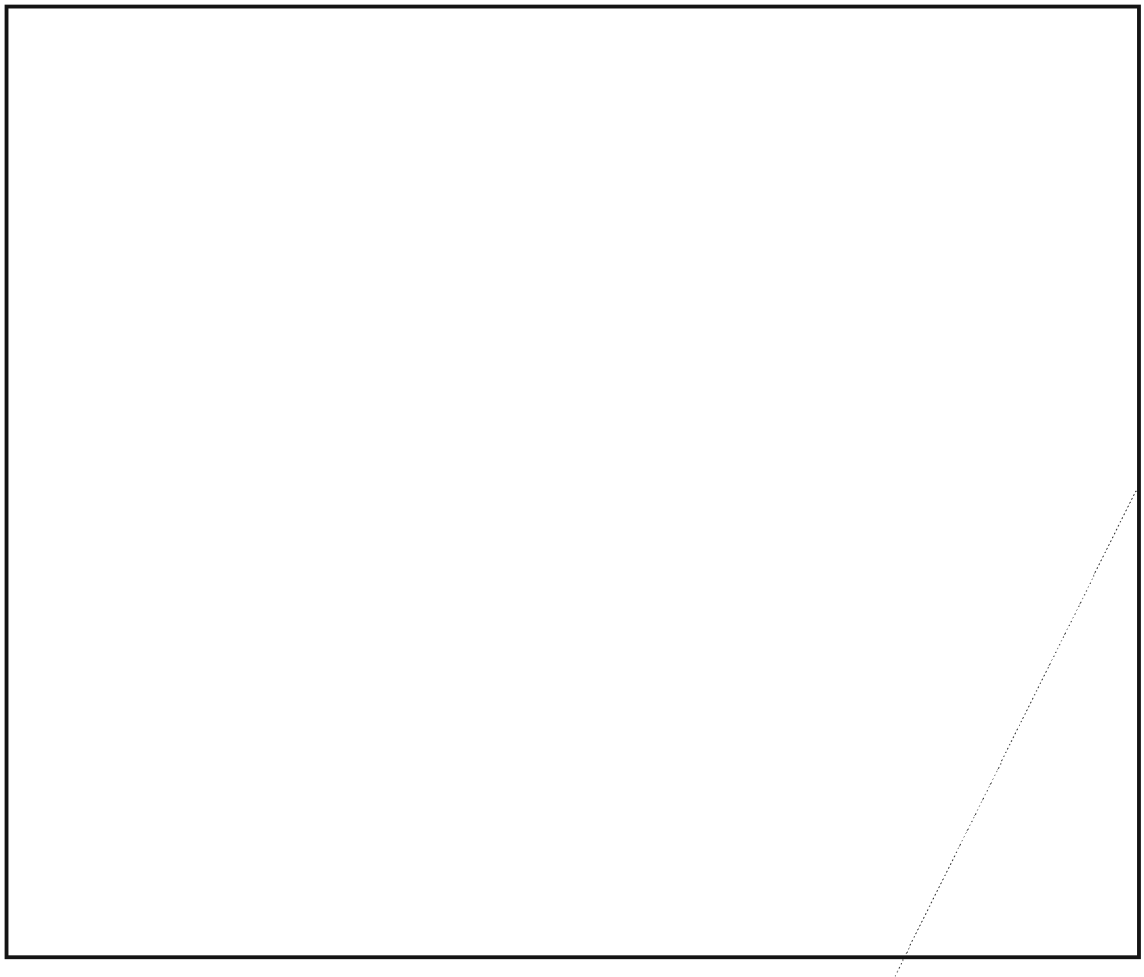
We assume that any computer or application built to these criteria would not be exportable. We also assume that at least for the first few offerings, the government will

have to underwrite a large share of the development costs because of the present relatively limited market for an A2 product.

We do not view A2 as the end of the line. A2 is but the next milestone on the difficult path to comprehensive computer security. There were several other additions to the criteria that we might have proposed, but we decided they constituted too great a step and decided to leave them for Classes A3 and beyond. Among them are compiler verification, more extensive hardware and software analysis, denial-of-service protection, etc. Denial-of-service is a particularly thorny problem. While some work has been done,[7] much more remains. The first step is to reach agreement as to what actually constitutes denial-of-service in a computer security context.

And finally, we do not view our proposal as the last word on the subject of what ought to constitute the A2 criteria. Our intention is but to begin the discussion, not to end it.

7. For the more formal treatments on the topic of denial of service, see: Virgil D. Gligor, "A Note on the Denial-of-Service Problem," *Proceedings, 1983 IEEE Symposium on Security and Privacy*, Silver Spring, Maryland: IEEE Computer Society Press, 1983, pp. 139–49, and Che-Fn Yu and Virgil D. Gligor, "A Formal Specification and Verification Method for the Prevention of Denial of Service," *Proceedings, 1988 IEEE Symposium on Security and Privacy*, Washington, D.C.: Computer Society Press of the IEEE, 1988, pp. 187–202.



(b)(3)-P.L. 86-36
(b)(6)